markovchain_package_simulation

Eduardo de Jesús Cuellar Chávez

4/28/2022

La paquetería `markovchain`

El día de hoy trabajaremos con la paquetería `markovchain` de R. Esta paquetería tiene integradas ciertas funciones que nos permiten simular características de las cadenas de Markov. La documentación completa la podemos consultar En este link

Instalación y carga de la paquetería

Lo primero será, evidentemente, instalar la paquetería:

```
#Descomentar y correr
#install.packages('markovchain')
```

Ahora, la cargamos:

```
library(markovchain)
```

```
## Package:  markovchain
## Version:  0.8.6
## Date:     2021-05-17
## BugReport: https://github.com/spedygiorgio/markovchain/issues
```

Aprendiendo a usar la paquetería a través de un ejemplo.

Para hacer más dinámico esto, planteemos un pequeño problema:

Pensemos en un conductor (taxista, uber, repartidor, etc) en la ciudad, la cual dividiremos en 4 zonas: Norte, sur, este y oeste. El conductor, cada que inicia un nuevo viaje, puede quedarse en su misma zona o ir a una distinta, dependiendo de cuál sea el destino al que debe de llegar, evidentemente.

¿Por qué es bueno pensar esto como una cadena de Markov? Porque la ubicación del taxista al tiempo siguiente dependerá únicamente de dónde se encuentre en ese momento: Si un repartidor va de la zona norte a la zona sur, al llegar a la zona sur, el haber estado en la zona norte no le afecta en los pedidos que pueda llegar a recibir, por ejemplo, solo importará que se encuentra en la zona sur en ese momento.

Ahora, supongamos que la dinámica de los viajes que realiza el conductor son los siguientes:

- Un 30% de los viajes en la zona norte son dentro de la misma zona, un 30% son para el sur y un 40% para el oeste.
- Un 40 % de los viajes en la zona sur son dentro de la misma zona, un 40% son hacia la zona norte y un 20% para la zona oeste
- Un 20% de los viajes en la zona oeste son dentro de la misma zona, un 50% son hacia la zona norte y un 30% hacia la zona sur.

- Un 70% de los viajes en la zona este son dentro de la misma zona, 20% son hacia la zona oeste y 10% a la zona sur.

¡Hagamos las matrices de transición!

Primero, creemos vectores con la probabilidad de transición, estando en un estado, hacia otro (Es decir, los vectores de $\mathbb{P}(x, y) \forall x, y \in \mathbb{S}$)

```
Zona_Norte = c(0.30, 0.30, 0.00, 0.40)
Zona_Sur = c(0.40, 0.40, 0, 0.20)
Zona_Este = c(0.00, 0.10, 0.70, 0.20)
Zona_Oeste = c(0.50, 0.30, 0.00, 0.20)

Probas_Zonas = c(Zona_Norte, Zona_Sur, Zona_Este, Zona_Oeste)
```

Creemos, también, un vector con los nombres de los espacios de estado:

```
Zonas = c("Norte", "Sur", "Este", "Oeste")
```

Creamos la matriz de transición:

```
Transicion_Zonas = matrix(data = Probas_Zonas,
                          nrow = length(Zonas),
                          ncol = length(Zonas),
                          byrow = TRUE,
                          dimnames =list(Zonas, Zonas))
Transicion_Zonas
```

```
##       Norte Sur Este Oeste
## Norte   0.3 0.3  0.0   0.4
## Sur     0.4 0.4  0.0   0.2
## Este    0.0 0.1  0.7   0.2
## Oeste   0.5 0.3  0.0   0.2
```

Ahora, creemos un objeto `markovchain`

```
MC_Zonas = new('markovchain',
               states = Zonas,
               byrow = TRUE,
               transitionMatrix = Transicion_Zonas,
               name = "Rutas_Conductor")
MC_Zonas
```

```
## Rutas_Conductor
##  A  4 - dimensional discrete Markov Chain defined by the following states:
##  Norte, Sur, Este, Oeste
##  The transition matrix  (by rows)  is defined as follows:
##       Norte Sur Este Oeste
## Norte   0.3 0.3  0.0   0.4
## Sur     0.4 0.4  0.0   0.2
## Este    0.0 0.1  0.7   0.2
## Oeste   0.5 0.3  0.0   0.2
```

Probas de n-pasos

Calculemos la probabilidad de que un conductor de la zona norte, permanezca en dicha zona despues de dos y tres viajes

Con las matrices normales, solo es:

```
Transicion_Zonas %*% Transicion_Zonas
```

```
##       Norte  Sur Este Oeste
## Norte  0.41 0.33 0.00  0.26
## Sur    0.38 0.34 0.00  0.28
## Este   0.14 0.17 0.49  0.20
## Oeste  0.37 0.33 0.00  0.30
```

```
Transicion_Zonas %*% Transicion_Zonas %*% Transicion_Zonas
```

```
##       Norte   Sur  Este Oeste
## Norte 0.385 0.333 0.000 0.282
## Sur   0.390 0.334 0.000 0.276
## Este  0.210 0.219 0.343 0.228
## Oeste 0.393 0.333 0.000 0.274
```

En general:

```
matriz_n_pasos <- function(transition_mtx, n){
  res = transition_mtx
  for (i in 1:(n-1)){
    res = res %*% transition_mtx
  }
  return(res)
}
```

```
matriz_n_pasos(Transicion_Zonas, 2)
```

```
##       Norte  Sur Este Oeste
## Norte  0.41 0.33 0.00  0.26
## Sur    0.38 0.34 0.00  0.28
## Este   0.14 0.17 0.49  0.20
## Oeste  0.37 0.33 0.00  0.30
```

```
matriz_n_pasos(Transicion_Zonas, 3)
```

```
##       Norte   Sur  Este Oeste
## Norte 0.385 0.333 0.000 0.282
## Sur   0.390 0.334 0.000 0.276
## Este  0.210 0.219 0.343 0.228
## Oeste 0.393 0.333 0.000 0.274
```

Con la paquetería, basta hacer lo siguiente:

```
MC_Zonas^2
```

```
## Rutas_Conductor^2
##  A  4 - dimensional discrete Markov Chain defined by the following states:
##  Norte, Sur, Este, Oeste
##  The transition matrix  (by rows)  is defined as follows:
##       Norte  Sur Este Oeste
## Norte  0.41 0.33 0.00  0.26
## Sur    0.38 0.34 0.00  0.28
## Este   0.14 0.17 0.49  0.20
## Oeste  0.37 0.33 0.00  0.30
```

```
MC_Zonas^3
```

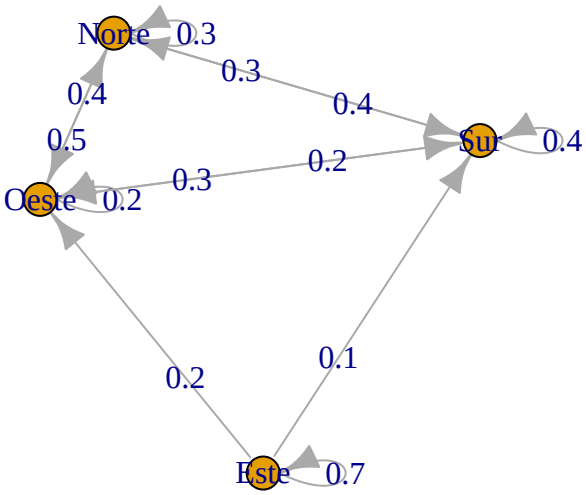
```
## Rutas_Conductor^3
```

```
##  A  4 - dimensional discrete Markov Chain defined by the following states:
##  Norte, Sur, Este, Oeste
##  The transition matrix  (by rows)  is defined as follows:
##        Norte   Sur  Este Oeste
## Norte 0.385 0.333 0.000 0.282
## Sur   0.390 0.334 0.000 0.276
## Este  0.210 0.219 0.343 0.228
## Oeste 0.393 0.333 0.000 0.274
```

Clasificación de estados

Podemos hacer un plot de la cadena también

```
plot(MC_Zonas)
```



¿Cuáles son los estados absorbentes?

```
recurrentStates(MC_Zonas)
```

```
## [1] "Norte" "Sur"   "Oeste"
```

¿Qué clases recurrentes hay?

```
recurrentClasses(MC_Zonas)
```

```
## [[1]]
## [1] "Norte" "Sur"   "Oeste"
```

¿Hay estados absorbentes?

```
absorbingStates(MC_Zonas)
```

```
## character(0)
```

¿Cuáles son los estados transitorios?

```
transientStates(MC_Zonas)
```

```
## [1] "Este"
```

¿Cuáles son las clases de estados transitorios?

```
transientClasses(MC_Zonas)
```

```
## [[1]]
```

```
## [1] "Este"
```

¿Cuáles son las clases que hay?

```
communicatingClasses(MC_Zonas)
```

```
## [[1]]
## [1] "Norte" "Sur"   "Oeste"
##
## [[2]]
## [1] "Este"
```

Probas de absobsión

Calculamos la probabilidad de que, estando en un estado transitorio, lleguemos a uno recurrente. Es una matriz de entradas (i,j) donde i (El renglón), son los estados transitorios, y j (Las columnas), son los estados recurrentes. nos devuelve la probabilidad de que el primer estado **no** transitorio al que pasemos, estando en i, sea el estado j, entrando así a la clase de comunicación a la que pertenece el estado j.

```
absorptionProbabilities(MC_Zonas)
```

```
##      Norte       Sur     Oeste
## Este     0 0.3333333 0.6666667
```

Calculamos el número esperado de pasos para ir de un estado transitorio a alguno recurrente. Nos devolverá un vector con el número esperado de pasos que necesita para llegar a un recurrente

```
meanAbsorptionTime(MC_Zonas)
```

```
##     Este
## 3.333333
```

Calculamos también, dada una cadena irreducible, el número esperado de pasos que se necesitan para alcanzar otros estados. Como la que elegimos no es irreducible, tendríamos que usar otra

```
#meanFirstPassageTime(MC_Zonas, "Norte")
```

Calculemos el número promedio de visitas. Nos devuelve una matriz con entradas (i,j), que representa el número promedio de visitas al estado j, si la cadena comienza en el estado i

```
meanNumVisits(MC_Zonas)
```

```
##       Norte Sur     Este Oeste
## Norte   Inf Inf 0.000000   Inf
## Sur     Inf Inf 0.000000   Inf
## Este    Inf Inf 2.333333   Inf
## Oeste   Inf Inf 0.000000   Inf
```

Calculemos el tiempo esperado que tardamos en regresar a un estado recurrente:

```
meanRecurrenceTime(MC_Zonas)
```

```
##    Norte      Sur    Oeste
## 2.571429 3.000000 3.600000
```

Calculemos la probabilidad de llegar por primera vez a un conjunto de estados A, antes que a un conjunto de estados B, comenzando desde diferentes estados:

```
committorAB(MC_Zonas, c(4), c(2), p = c(3))
```

```
##      Este
## 0.6666667
```

Recuperamos las probas de cada estado

```
conditionalDistribution(MC_Zonas, "Norte")
```

```
## Norte   Sur  Este Oeste
##   0.3   0.3   0.0   0.4
```

Calculemos las probabilidades de que, comenzando en el estado i, lleguemos alguna vez al estado j

```
hittingProbabilities(MC_Zonas)
```

```
##       Norte Sur Este Oeste
## Norte     1   1  0.0     1
## Sur       1   1  0.0     1
## Este      1   1  0.7     1
## Oeste     1   1  0.0     1
```

Veamos si una cadena es irreducible

```
is.irreducible(MC_Zonas)
```

```
## [1] FALSE
```

Calculemos la probabilidad de que la primera visita al estado j, comenzando desde el estado que fijamos, sea en el paso i (Nos devuelve una matriz de dimensiones i,j)

```
firstPassage(MC_Zonas, "Norte", 10)
```

```
##          Norte        Sur Este      Oeste
## 1  0.300000000 0.30000000    0 0.40000000
## 2  0.320000000 0.21000000    0 0.18000000
## 3  0.166000000 0.14700000    0 0.12600000
## 4  0.093200000 0.10290000    0 0.08820000
## 5  0.052600000 0.07203000    0 0.06174000
## 6  0.029696000 0.05042100    0 0.04321800
## 7  0.016765600 0.03529470    0 0.03025260
## 8  0.009465440 0.02470629    0 0.02117682
## 9  0.005343952 0.01729440    0 0.01482377
## 10 0.003017062 0.01210608    0 0.01037664
```

Para un subconjunto de estados:

```
firstPassageMultiple(MC_Zonas, "Norte", c("Norte", "Sur"), 10)
```

```
##           set
## 1  0.60000000
## 2  0.53000000
## 3  0.31300000
## 4  0.19610000
## 5  0.12463000
## 6  0.08011700
## 7  0.05206030
## 8  0.03417173
## 9  0.02263836
## 10 0.01512314
```

```
steadyStates(MC_Zonas)
```

```
##          Norte       Sur Este     Oeste
## [1,] 0.3888889 0.3333333    0 0.2777778
```